Planning maximum-manipulability cutting paths

T. Pardi, V. Ortenzi, C. Fairbairn, T. Pipe, A. M. Ghalamzan-E., and R. Stolkin

***Abstract*—This paper presents a method for constrained motion planning from vision, which enables a robot to move its end-effector over an observed surface, given start and destination points. The robot has no prior knowledge of the surface shape, but observes it from a noisy point cloud. We consider the multi-objective optimisation problem of finding robot trajectories which maximise the robot's manipulability throughout the motion, while also minimising surface-distance travelled between the two points. This work has application in industrial problems of *rough* robotic cutting, *e.g.*, demolition of legacy nuclear plant, where the cut path needs not be precise as long as it achieves dismantling. We show how detours in the path can be leveraged to increase the manipulability of the robot at all points along the path. This helps to avoid singularities, while maximising the robot's capability to make small deviations during task execution. We show how a sampling-based planner can be projected onto the Riemannian manifold of a curved surface, and extended to include a term which maximises manipulability. We present the results of empirical experiments, with both simulated and real robots, which are tasked with moving over a variety of different surface shapes. Our planner enables successful task completion, while ensuring significantly greater manipulability when compared against a conventional RRT* planner.**

***Index Terms*—Motion and Path Planning, Kinematics, Robotics in Hazardous Fields**

I. Introduction

ROBOTIC cutting involves an interesting problem of path-planning for a serial arm under semi-closed chain constraints. The end-effector cutting tool is constrained to touch the cutting surface, thereby forming a closed chain at any given time step. However, the cutting surface can be regarded as a manifold upon which the end-effector has *locally* two or three degrees of freedom to move, *e.g.*, cutting with rotary tools. The problem of our interest is *rough* cutting, *e.g.* in robotic demolition in hazardous environments. In such applications, the exact cutting path is not important, as long as the robot successfully, *e.g.*, cuts an object into two pieces, or cuts open a container to inspect its contents. Here we therefore address the cutting-path planning for a serial manipulator.

Fig. 1. Nuclear decommissioning worker wearing an air-fed plastic suit underneath a heavy leather overcoat, and multiple layers of gloves, while using power tools to cut legacy nuclear plant contaminated by alpha-radiation emitting substances, such as plutonium dust. The leather coat protects the plastic suit from being punctured by hot sparks during cutting. Maximum 2hrs work per day is possible, due to extreme discomfort and heat exhaustion as the suit fogs and fills with sweat. Image courtesy of Sellafield Ltd.

Rough cutting is a key element for using robots for cleanup of legacy nuclear waste [1], [2]. This is an international challenging problem in more than 30 countries with a nuclear history. The UK alone contains an estimated 4.9 million tonnes of legacy nuclear waste [3], dating back to the 1950s. The UK legacy cleanup is expected to take 120 years, at a cost of order $300billion. Numerous ageing and disused buildings contain contaminated plant, vessels, pipework and "cell furniture" which must be dismantled. All waste must be "size-reduced" to fit the maximum waste into the minimum expensive storage containers which must each be stored and monitored for many years at great cost. It is also necessary to cut highly contaminated material (*e.g.*, a hot spot in a pipe) away from more benign material - this saves filling extremely expensive high-level waste containers with low hazard material. Without significant advances in robotics, it is expected that the UK cleanup will require one million entries of human workers, wearing air-fed plastic suits, into hazardous zones, Fig. 1. However, in many cases, radiation levels are too high to permit any entry of humans, even with protective suits. Robotic demolition methods will be essential in such cases.

A variety of tooling can be used for robotic cutting. Our team previously worked closely with the UK nuclear industry to achieve a world-first autonomous vision-guided robotic laser cutting of contaminated metal inside a radioactive facility [4].

Manuscript received: September, 10, 2019; Revised November, 7, 2019; Accepted January, 15, 2020.

This paper was recommended for publication by Editor Youngjin Choi upon evaluation of the Associate Editor and Reviewers' comments. Tommaso Pardi is supported by a doctoral bursary of the UK Nuclear Decommissioning Authority. Rustam Stolkin is supported by a Royal Society Industry Fellowship. This work forms part of the UK National Centre for Nuclear Robotics initiative, part-funded by EPSRC EP/R02572X/1. The work was also partially supported by Faraday ReLiB, EPSRC EP/P017487/1, EP/P01366X/1.

T. Pardi, V. Ortenzi, and R. Stolkin are with the Extreme Robotics Lab (ERL), University of Birmingham, UK {txp754, v.ortenzi, R.Stolkin}@bham.ac.uk

T. Pipe is with the Bristol Robotics Lab (BRL), University of West England, UK Anthony.Pipe@uwe.ac.uk

C. Fairbairn is with the National Nuclear Laboratory, UK Anthony.Pipe@uwe.ac.uk

A. Ghalamzan is with Intelligent Manipulation Lab and Lincoln Centre for Autonomous System, University of Lincoln, UK, aghalamzanesfahani@lincoln.ac.uk

T. Pardi, V. Ortenzi, C. Fairbairn, T. Pipe, and R. Stolkin are with the UK National Centre for Nuclear Robotics

Digital Object Identifier (DOI): see top of this page.

Lasers, and other non-contact methods such as water jet or plasma cutting, are convenient in that no contact forces are exerted, although close geometric surface following (with a few millimetres stand-off) must still be achieved.

In contrast, we are now considering the use of axial rotary cutting tools (similar to a milling machine cutter). Kinematic path-planning constraints for such tools are similar to those for a laser: the cutter axis must be maintained normal to local surface curvature, and rotations of the robot around the tool axis are allowable. However, with the rotary cutter, forceful interactions between the robot and cut materials of uncertain properties, introduce significant perturbations. We would like the robot to have sufficient manipulability to provide capacity for responding compliantly to such perturbations, while following a cutting path between two points given by a human operator. Maximising manipulability [5] also results in motion plans which will not cause any joints of the robot to pass near singularity configurations.

This paper shows how to plan serial manipulator cutting paths, on smooth but otherwise arbitrarily shaped surfaces, observed as a noisy Point Cloud (PC). We show how points from the PC can be used to generate a net of possible path nodes, by mapping them from Euclidean space onto a Riemannian manifold in which sampling-based path-planning takes place. We further show how a cost function can be constructed, which considers kinematics and reachability constraints, while evaluating manipulability of the robot throughout its motion along any candidate path. The main contributions of this paper are as follows: (1) we propose a procedure to project observed points from the PC onto the Riemannian manifold of the object; (2) we design a bi-objective cost function, which maximises manipulability, while also reducing overall path length during path optimisation; (3) we incorporate the cost function into an RRT* planner; (4) we empirically evaluate the planner with both real and simulated robots, on a variety of surface shapes.

This paper is structured as follows. Section II explains our work in the context of the robot path-planning literature. We formulate the problem in Section III, describing the manipulability index and Riemannian manifold. Section IV shows how the concepts from Section III can be incorporated into a modified RRT* algorithm, which computes a path on a Riemannian surface observed as a noisy PC, while maximising manipulability along the path. In Section V, we demonstrate the effectiveness of our modified RRT* cut planner, in experiments using simulated and real robots. Our proposed planner yields cut paths with higher manipulability as compared to a conventional RRT* approach.

II. Related work

There is comparatively little literature from the robotics research community on cutting. Recent work on robotic tool use includes *e.g.* [6], in which a co-bot learns to assist a human with a backwards and forwards sawing motion. However, this work does not consider actually planning a cutting path. There is a large body of literature on path-planning for tool paths in multi-axis CNC machining [7]. Such work is aimed at precision manufacturing, where it is essential that the cutting tool rigidly follows an exact path through the work-piece with a known geometrical model.

In contrast, for *rough* cutting in *e.g.* nuclear decommissioning or disaster response, we are not concerned about following exact cut paths. Instead, it is much more important for us to consider safety of the manipulator's kinematics throughout the motion. We therefore modify the cutting path itself, to avoid singularity configurations for the arm, and to improve robustness to perturbations by maximising manipulability throughout the motion [8], [9]. Furthermore, our applications involve highly unstructured environments, Fig. 1. Thus, we must plan cuts on arbitrary objects observed by noisy partial PC views.

There are well established robust methods for robot path-planning with obstacle avoidance. Early work by Khatib modelled obstacles as artificial potential fields, and optimised collision-free paths by descending an energy gradient, [10]. More recent methods for path-planning by gradient descent of cost functions, include the well-known CHOMP [11] and STOMP [12]. However, the computation complexity makes such approaches not suitable for close to real-time applications. On the other hand, it is now common to use sampling-based methods, such as PRM [13], RRT [14], and RRT* [15], [16], to generate a net of points which can be explored to find collision-free paths faster than STOMP and CHOMP. However, unmodified conventional path planning algorithms are not immediately useful for computing a cutting path on an object surface suitable for a robotic manipulator. An end-effector (cutting tool) path computed with these approaches may be out of the reachable workspace of the manipulator or may pass through singular configurations of the robot.

Various authors have sought to augment the classical path-planning approaches by incorporating modified cost functions, based on additional information, to induce useful supplementary robotic behaviours. A cost-based optimisation approach was proposed in [17], to enable an Autonomous Underwater Vehicle to plan a path between specified start and destination locations. The robot plans large deviations in its route, to avoid adverse currents, while exploiting currents in useful directions to minimise energy expenditure during the journey. Related Unmanned Aerial Vehicle literature also considers environmental factors, *e.g.* [18]; and planning on non-flat surfaces with constraints is considered in legged locomotion [19]. However, the constraints imposed by a manipulator's kinematics are different from those of mobile robots.

[20] generates paths on surfaces using a sample-based approach. In contrast with our work, an a-priori mathematical description of the surface is assumed. During cutting, a serial arm is constrained to form a closed chain with the cut surface. A smaller body of work involves planning of the end-effector under constraints, [21], [22].

In [23] and [24], a transition-based RRT algorithm, driven by a work-based cost, generates a collision-free path between points. In [25], a heuristically biased RRT is proposed to guide the search on the tree. Inverse kinematics guides an RRT-based search [26], while the manipulability of the robot is exploited to bias the sampling process in [27]. Our previous work [8], [9] proposed a metric for evaluating manipulability throughout

an entire trajectory, and showed how this could be applied for grasp planning. Here we extend these ideas to cut-path planning.

III. Problem formulation

We generate paths from point A to point B using RRT*. To cope with noisy point clouds of objects, we incorporate *Riemannian manifold mapping* to generate samples on a smooth surface. We introduce a cost function, based on the manipulability of the robot, into RRT* to address possible kinematics issues throughout motions. These modifications to the naive RRT* guarantee that the computed path (i) connects the start point (A) and the endpoint (B) (possibly specified by a user), (ii) lies on the object surface and (iii) is feasible for the manipulator. We call our approach "RRT*-RMM" (RRT* with Riemannian Manifold mapping and Manipulability cost).

Rapidly-exploring Random Tree:* Rapidly-exploring Random Tree (RRT) is one of the most common sampling-based path planners, [14]. The basic idea behind RRT is to sample points within a space of interest, *e.g.* either in Cartesian or configuration space, and add them in a tree structure based on a distance metric. At every iteration, the algorithm generates a new point taking into account constraints imposed by, *e.g.*, the robot kinematics. Then, it connects that point to the closest node in the tree. RRT* [15] is an extension to the classical RRT, which allows the re-evaluation of nodes already in the tree when a new point is available. This procedure is usually referred to as *rewiring*. During the rewiring, the algorithm selects the neighbourhood of a point (points in the tree within a range distance to the point) and evaluates whether these nodes improve their value passing through the new available point. This process provides RRT with better convergence to a solution and the solution converges to the shortest path as the number of samples goes to infinity.

Manipulability: Let $\boldsymbol{q} \in \mathbb{R}^n$ represent the robot configuration where n is the number of degrees of freedom (dof) of the robot. Given a specific $\boldsymbol{q}$, position and orientation of every point of the robot are uniquely defined (forward kinematics). This mapping, $\boldsymbol{f_r}$, is commonly expressed as

$$\boldsymbol{r} = \boldsymbol{f_r}(\boldsymbol{q}), \tag{1}$$

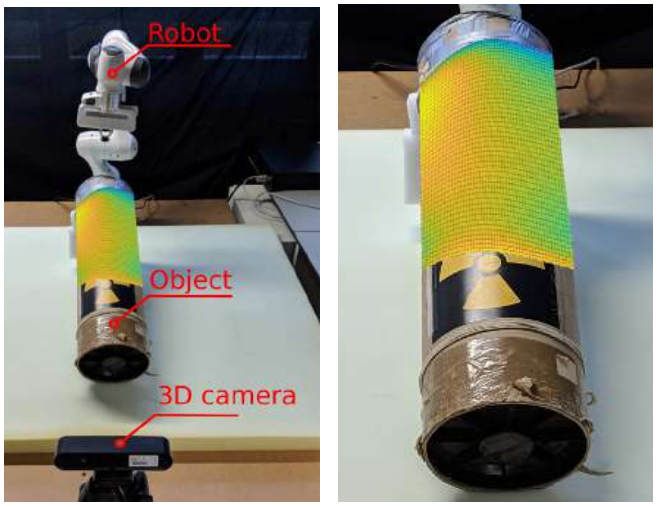
where $\boldsymbol{r} \in \mathbb{R}^m$ is the position and/or the orientation of a point of interest of the robot in the Cartesian space and m is the dimension of this representation (*e.g.*, $m = 3$ for 3D position, $\boldsymbol{r} = [\boldsymbol{p}]$; or $m = 6$ for 3D position ($\boldsymbol{p}$) and Euclidean orientation ($\boldsymbol{\psi}$), $\boldsymbol{r} = [\boldsymbol{p} \quad \boldsymbol{\psi}]^T$)). Differential kinematics are defined using the robot Jacobian $\boldsymbol{J(q)}$ as

$$\dot{\boldsymbol{r}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2}$$

and relate velocities in the configuration space to velocities in the Cartesian space[1]. If we constrain the norm of the configuration velocities to be unitary, the configuration lies on the unitary sphere $\mathbb{S}^1$

$$|\dot{\boldsymbol{q}}| = \dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}^T\boldsymbol{J}^{\dagger T}\boldsymbol{J}^{\dagger}\dot{\boldsymbol{r}} = \dot{\boldsymbol{r}}^T\boldsymbol{\Gamma}^{\dagger}\dot{\boldsymbol{r}} = 1 \tag{3}$$

[1]Since the Jacobian matrix always depends on the configuration $\boldsymbol{q}$, we drop the dependence on $\boldsymbol{q}$, and in the following we write $\boldsymbol{J(q)}$ as $\boldsymbol{J}$.



(a) (b)

Fig. 2. An experimental robotic cutting setup (Fig. 2(a)). A 3-D camera (positioned in front of the robot at 2.8 [m] distance facing the robot) captures the point cloud of the object surface. Our approach computes a cutting path with given initial and end points. This path is suitable for the robot kinematics as our algorithm accounts for a manipulability index. Fig. 2(b) shows the manipulability of the robot: each point is coloured based on the manipulability corresponding to the configuration the robot is in when its end effector touches such point. Points with lower manipulability are shown in blue; points with higher manipulability are represented in yellow and red.

where † is the inverse matrix whether $\boldsymbol{J}$ is square or the pseudo-inverse matrix otherwise. Previous work leverages manipulability to yield optimal manipulation movements for planning a suitable grasping pose [9]. We would also like to optimise the manipulation capability for robotic cutting. The conventional measure of manipulability [28] is defined as

$$w(\boldsymbol{q}) = \sqrt{det(\boldsymbol{\Gamma})} = \sqrt{\lambda_1\lambda_2...\lambda_n}, \tag{4}$$

where λ_i are the eigenvalues of $\boldsymbol{\Gamma}$. This index provides a value that is proportional to the volume of the manipulability ellipsoid, and it does not require a long computational time.

Riemannian Manifold: A manifold is an n-topological space that approximates the Euclidean space in the neighboorhood of its points [29]. A smooth manifold is a differentiable manifold for which all the transition maps are smooth. That is, derivatives of all orders exist; so it is a C^k-manifold for all k.

A Riemannian manifold is a pair $[\mathcal{M}, z]$ where $\mathcal{M}$ is a smooth manifold and z is an inner product of two vector spaces on the manifold. The family of these inner products represents the Riemannian metric.

In Fig. 3, a smooth manifold $\mathcal{M}$ is shown using red colour, and in blue the figure shows the tangent plane to $\mathcal{M}$ at a point, $\boldsymbol{p} \in \mathcal{M}$, namely $T_p\mathcal{M}$.

By definition of Riemannian manifold, given a point $\boldsymbol{p}_T \in T_p\mathcal{M}$ and a manifold $\mathcal{M}$, as per Fig. 3, a function which maps the point $\boldsymbol{p}_T$ onto the manifold $\mathcal{M}$ exists [29]. Applying the mapping to the point $\boldsymbol{p}_T$, we obtain a point $\boldsymbol{p}_{\mathcal{M}} \in \mathcal{M}$, which is the projection of $\boldsymbol{p}_T$ onto $\mathcal{M}$. The red vector connecting $\boldsymbol{p}$ and $\boldsymbol{p}_{\mathcal{M}}$ in Fig. 3 is unique, and it represents the shortest travelling distance between the two points lying onto the manifold, namely the geodesic distance. The map between points onto

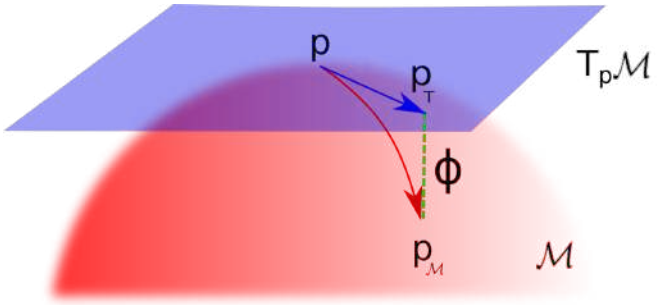


Fig. 3. Given a smooth manifold $\mathcal{M}$, showed in red, the neighbourhood of every point $\boldsymbol{p}$ onto the manifold can be approximated with a tangent plane $T_p\mathcal{M}$, coloured in blue. The function ϕ uniquely maps every point $\boldsymbol{p}' \in T_p\mathcal{M}$ onto the manifold $\mathcal{M}$, namely $\boldsymbol{p}''$.

the plane $T_p\mathcal{M}$ and points onto the manifold $\mathcal{M}$ is called *exponential* map, $\phi : T_p\mathcal{M} \mapsto \mathcal{M}$, and it is defined as per eq. (5).

$$\boldsymbol{p}_{\mathcal{M}} = \phi(\boldsymbol{p}_T) = \exp_p(\boldsymbol{p}_T). \tag{5}$$

Conversely, the inverse mapping that projects point belonging to the manifold, $\mathcal{M}$, onto the tangent plane, $T_p\mathcal{M}$, is called *logaritmic* map, $\phi^{-1} : \mathcal{M} \mapsto Tp_{\mathcal{M}}$.

$$\boldsymbol{p}_T = \phi^{-1}(\boldsymbol{p}_T) = \log_p(\boldsymbol{p}_{\mathcal{M}}) \tag{6}$$

IV. Proposed method RRT*-RMM

Given a Point Cloud of an object, which includes a set of points, *i.e.*, $\mathbb{P} = \{\boldsymbol{p}_{\mathbb{P}}^1, ..., \boldsymbol{p}_{\mathbb{P}}^m\}$, where $\boldsymbol{p}_{\mathbb{P}} \in \mathbb{R}^3$ and m is the number of points in the set, we need to compute a series of points representing a cutting path between a given initial point $\boldsymbol{p}_{\mathbb{P}}^A \in \mathbb{P}$, and a given terminal point $\boldsymbol{p}_{\mathbb{P}}^B \in \mathbb{P}$. We use standard inverse kinematics to obtain the joint space configuration, $\boldsymbol{q}^A$, corresponding to $\boldsymbol{p}_{\mathbb{P}}^A$. We assume a PC captured by a depth sensor is a noisy data of a smoothly curved object, *i.e.*, $\boldsymbol{p}_{\mathbb{P}} = \hat{\boldsymbol{p}}_{\mathcal{M}} + \sigma$, where $\sigma \in \mathbb{R}^3$ is white noise and $\hat{\boldsymbol{p}}_{\mathcal{M}}$ is a point sampled from the manifold. We assume the set of points $\hat{\boldsymbol{p}}_{\mathcal{M}}$ to represent (discretely) a smooth (Riemannian) manifold. We aim to build a tree structure whose nodes represent a path suitable for a cutting task. Every node of the tree η includes a robot's end-effector (EE) pose suitable for cutting, $\boldsymbol{r} = [\boldsymbol{p}_{\mathcal{M}},\ \boldsymbol{\psi}]^T$ where $\boldsymbol{p}_{\mathcal{M}}$ and $\boldsymbol{\psi}$ denote the corresponding position and orientation, and the corresponding robot configuration $\boldsymbol{q}$, *i.e.* $\eta = \{\boldsymbol{p}_{\mathcal{M}}, \boldsymbol{\psi}, \boldsymbol{q}\}$. The desired tree includes a series of points $\boldsymbol{\xi} = \{\boldsymbol{p}_{\mathcal{M}}^1, \boldsymbol{p}_{\mathcal{M}}^2, ..., \boldsymbol{p}_{\mathcal{M}}^n\}$, where $\boldsymbol{p}_{\mathcal{M}}^i \in \mathcal{M}$, and $\boldsymbol{\xi}$ connects the desired initial, $\boldsymbol{p}_{\mathcal{M}}^1 = \boldsymbol{p}_{\mathbb{P}}^A$, and terminal points, $\boldsymbol{p}_{\mathcal{M}}^n = \boldsymbol{p}_{\mathbb{P}}^B$. We assume the orientation of the end-effector is determined by a cutting task, *e.g.*, for cutting with a knife, the EE must be normal to $\mathcal{M}$ and the the knife needs to cut the object moving from $\boldsymbol{p}_{\mathcal{M}}^{i-1}$ to $\boldsymbol{p}_{\mathcal{M}}^i$.

At the *j-th* iteration of our algorithm, a point from the PC is randomly selected, $\boldsymbol{p}_{\mathbb{P}}^{rand} \in \mathbb{P}$. Then, the algorithm finds a tree's vertex, $\eta^j = \{\boldsymbol{p}_{\mathcal{M}}^{nearest}, \boldsymbol{\psi}, \boldsymbol{q}\}$, whose point, $\boldsymbol{p}_{\mathcal{M}}^{nearest}$, has the minimum distance to $\boldsymbol{p}_{\mathbb{P}}^{rand}$. Because we do not have a mathematical expression of the object surface, which can be represented by $\mathcal{M}$, we cannot compute a plane tangent to $\mathcal{M}$. However, we can approximate the tangent plane at point $\boldsymbol{p}_{\mathcal{M}}^{nearest}$, namely $T_p\mathcal{M}$, by applying principal component analysis (PCA) on the points in the close neighbourhood of $\boldsymbol{p}_{\mathcal{M}}^{nearest}$. We now project the point $\boldsymbol{p}_{\mathbb{P}}^{rand}$ onto $T_p\mathcal{M}$, we name the projected point as $\boldsymbol{p}_T \in T_p\mathcal{M}$. Then, we compute the point $\boldsymbol{p}_T^{\beta}$ using the projected and nearest points as per eq. 7 by choosing a very small value for the step size β. This ensures the updated value on the tangent plane will be correctly mapped onto the Riemannian manifold through the exponential mapping in the next steps of the algorithm.

$$\boldsymbol{p}_T^{\beta} = \boldsymbol{p}_{\mathcal{M}}^{nearest} + \beta(\boldsymbol{p}_T - \boldsymbol{p}_{\mathcal{M}}^{nearest}). \tag{7}$$

Fig. 4 summarises the steps for projecting points from the PC to the manifold. This figure shows a sample PC, the underlying smooth manifold in red, and the tangent plane to $\boldsymbol{p}_{\mathcal{M}}^{nearest}$, $T_p\mathcal{M}$, is shown with blue colour. In Fig. 4(b), $\boldsymbol{p}_T$ is the projection of $\boldsymbol{p}_{\mathbb{P}}^{rand}$ onto $T_p\mathcal{M}$. We choose β in eq. 7 to be a positive definite value with $\beta \ll 1$, this assures us that the projected point on the tangent plane, $\boldsymbol{p}_T^{\beta}$, is very close to $\boldsymbol{p}_{\mathcal{M}}^{nearest}$, Fig. 4 (c), which satisfies the underlying assumption for exponential mapping from tangent plane to the Riemannian manifold, $\mathcal{M}$, [29]. Note that $\boldsymbol{p}_{\mathcal{M}}^{nearest}$, $\boldsymbol{p}_T^{\beta}$, and $\boldsymbol{p}_T$ lie on the tangent plane, Fig. 4 (c). Finally, we obtain a new point, $\boldsymbol{p}_{\mathcal{M}}^{new} \in \mathcal{M}$ using exponential mapping, as per eq. (5). In order to account for the kinematic of the manipulator in our cutting path planning, we introduce a modified cost to be used in our RRT* algorithm. We use (1) a cost accounting for the distance, denoted by C_d, and (2) a cost for manipulability, denoted by $C_{\boldsymbol{M}}$.

As for C_d, we compute the sum of all segments over the path to reach $\boldsymbol{p}_{\mathcal{M}}^{new}$ starting from the root of the tree, $\boldsymbol{p}_{\mathcal{M}}^1$. That is the sum of Euclidean distances between consecutive points in the tree $\{\bar{\boldsymbol{p}}_{\mathcal{M}}^1, \bar{\boldsymbol{p}}_{\mathcal{M}}^2, ..., \bar{\boldsymbol{p}}_{\mathcal{M}}^I\}$, which is the sequence of points from the root of the tree to the point $\boldsymbol{p}_{\mathcal{M}}^{new}$, as per eq. (8).

$$C_d(\boldsymbol{p}_{\mathcal{M}}^{new}) = g(\bar{\boldsymbol{p}}_{\mathcal{M}}^I, \boldsymbol{p}_{\mathcal{M}}^{new}) + \sum_{i=1}^{I} g(\bar{\boldsymbol{p}}_{\mathcal{M}}^i, \bar{\boldsymbol{p}}_{\mathcal{M}}^{i-1}) \tag{8}$$

where I is the number of points visited in the tree before reaching $\boldsymbol{p}_{\mathcal{M}}^{new}$ and $g(.)$ is the geodesic distance between two adjacent points in the path. Assuming adjacent points are very close, this geodesic distance can be approximated by $g(\boldsymbol{p}^i, \boldsymbol{p}^{i-1}) = \|\boldsymbol{p}^i - \boldsymbol{p}^{i-1}\|$.

The manipulability cost is also computed over the path, as per eq. (9).

$$C_{\boldsymbol{M}}(\boldsymbol{q}^{new}) = \frac{1}{I+1}\left(\frac{1}{w(\boldsymbol{q}^{new})} + \sum_{i=1}^{I}\frac{1}{w(\boldsymbol{q}^i)}\right) \tag{9}$$

where $\boldsymbol{q}^{new} = \boldsymbol{q}^{nearest} + \boldsymbol{J}^{\dagger}(\boldsymbol{r}^{new} - \boldsymbol{r}^{nearest})$ is the corresponding configuration computed using the differential kinematics to match the displacement between $\boldsymbol{p}_{\mathcal{M}}^{nearest}$ and $\boldsymbol{p}_{\mathcal{M}}^{new}$, $\boldsymbol{q}^1 = \boldsymbol{q}^A$, and w is the manipulability index presented in eq. (4). We use the inverse of w so that minimising the manipulability cost is equivalent to maximising manipulability.

Eventually the cost is a weighted sum of $C_{\boldsymbol{M}}$ and C_d, as per eq. (10).

$$C(\boldsymbol{p}_{\mathcal{M}}^{new}, \boldsymbol{q}^{new}) = (1-\alpha)C_d(\boldsymbol{p}_{\mathcal{M}}^{new}) + \alpha C_{\boldsymbol{M}}(\boldsymbol{q}^{new}) \tag{10}$$

The coefficient $\alpha \in [0, 1]$ sets a trade-off between $C_{\boldsymbol{M}}$ and C_d. The proposed algorithm turns into the classical RRT*

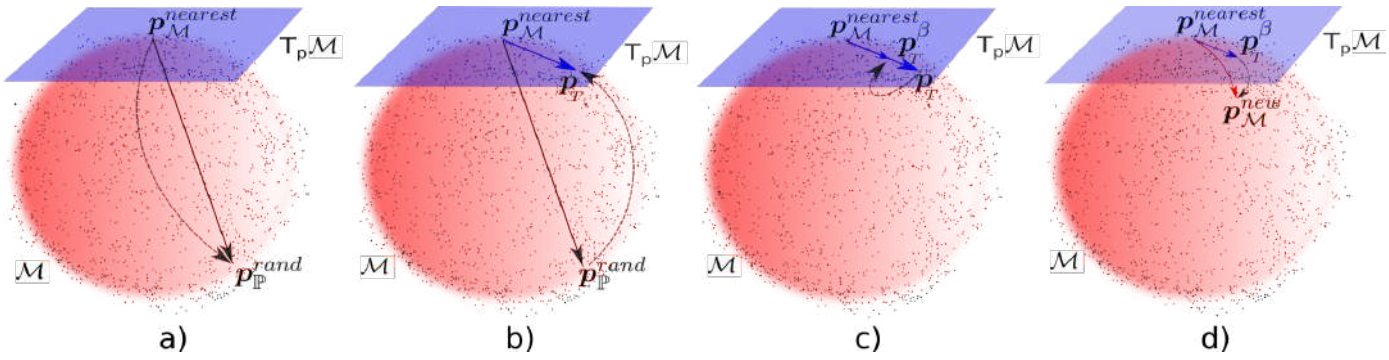


Fig. 4. This image shows the procedure to obtain a new point on the manifold from a point on the Point Cloud (PC). We sample a point from the PC, $\mathbb{P}$, and obtain $\boldsymbol{p}_{\mathbb{P}}^{rand}$, Fig.4(a). We project $\boldsymbol{p}_{\mathbb{P}}^{rand}$ onto the tangent manifold $T_p\mathcal{M}$ and obtain $\boldsymbol{p}_T$, Fig.4(b). Then, we take a step starting from $\boldsymbol{p}_{\mathcal{M}}^{nearest}$ towards $\boldsymbol{p}_T$ using β, and obtain $\boldsymbol{p}_T^{\beta}$, Fig.4(c). Finally, Fig.4(d) shows the mapping of the point $\boldsymbol{p}_T^{\beta} \in T_p\mathcal{M}$ to the manifold $\mathcal{M}$ using the map ϕ, thus obtaining $\boldsymbol{p}_{\mathcal{M}}^{new}$.

Algorithm 1 RRT*-Riemannian-mapping-manipulability

Param: $\eta = \{\boldsymbol{r},\ \boldsymbol{q}\}$, is tree node,
  $\boldsymbol{r} = [\boldsymbol{p}_{\mathcal{M}},\ \boldsymbol{\psi}]$ is end-effector pose on $\mathcal{M}$
tree.init(η^1)
for $k = 1$ *to* K **do**
  $\boldsymbol{p}_{\mathbb{P}}^{rand} \leftarrow getPointFromPntCloud()$
  $\boldsymbol{r}^{nearest},\ \boldsymbol{q}^{nearest} \leftarrow Nearest(\boldsymbol{p}_{\mathbb{P}}^{rand})$
  $T_p\mathcal{M} \leftarrow computeTangentPlane(\boldsymbol{p}_{\mathcal{M}}^{nearest})$
  $\boldsymbol{p}_T \leftarrow projectOnTangentPlane(\boldsymbol{p}_{\mathbb{P}}^{rand}, T_p\mathcal{M})$
  $\boldsymbol{p}_T^{\beta} \leftarrow takeAStep(\boldsymbol{p}_T, \boldsymbol{p}_{\mathcal{M}}^{nearest})$
  $\boldsymbol{p}_{\mathcal{M}}^{new} \leftarrow expRiemannianMap(\boldsymbol{p}_T^{\beta})$
  $\boldsymbol{\psi}^{new} \leftarrow getTangentPlaneNormal(T_p\mathcal{M})$
  $\boldsymbol{r}^{new} \leftarrow [\boldsymbol{p}_{\mathcal{M}}^{new},\ \boldsymbol{\psi}^{new}]^T$
  $\boldsymbol{q}^{new} \leftarrow \boldsymbol{q}^{nearest} + J^{\dagger}(\boldsymbol{q}^{nearest})(\boldsymbol{r}^{new} - \boldsymbol{r}^{nearest})$
  if $ObstacleFree(\boldsymbol{q}^{new})$ **then**
    $\boldsymbol{R}^{near} \leftarrow getSetofNearVertices(\boldsymbol{r}^{new})$
    for $h = 1$ *to* $H = |\boldsymbol{R}^{near}|$ **do**
      $C_d \leftarrow computeDistance(\boldsymbol{r}^{new}, \boldsymbol{R}^{near}(h))$
      $C_{\boldsymbol{M}} \leftarrow computeMan(\boldsymbol{r}^{new}, \boldsymbol{R}^{near}(h))$
      $\boldsymbol{C}(h) = (1-\alpha)C_d + \alpha C_{\boldsymbol{M}}$
    $\boldsymbol{r}^{best} \leftarrow selectMinimumElement(\boldsymbol{C})$
    tree.addVertex(η^{new})
    tree.addEdge(η^{best}, η^{new})
    tree.rewire($\boldsymbol{R}^{near}$, η^{new})
return tree

when $\alpha \to 0$; while $\alpha \to 1$ means only the manipulability is considered for planning the cutting path. This parameter must be chosen based on the corresponding domain knowledge, *e.g.*, we may need a different value of α for cutting with a knife or for cutting with a rotatory tool.

V. Experimental results and discussion

We use a Panda robot manufactured by Franka EMIKA for the real-world experiments[2]. We also provide some results with Sawyer in V-REP[3]. Although, both Panda and Sawyer are 7-DOF robotic arms equipped with a standard parallel jaw gripper, they have different kinematic chain. This shows how the proposed approach is readily transferable across different manipulator's kinematics. An Orbbec Astra RGB-D camera scans the area in front of the robot (Fig. 2), and we remove points outside the robot's workspace as preprocessing filtering on the point cloud. The camera is calibrated with respect to the robot base frame. As such, we can express the PC captured by the camera in the robot base frame. Furthermore, we attach two markers to each object, as shown in Fig. 5(a). These markers represent the start point A and the end point B of the path and allow a human operator to select the initial and end points of the cut. The RGB-D camera takes the point cloud of the scene in front of the robot as input, and our algorithm computes a cutting path between point A and point B.

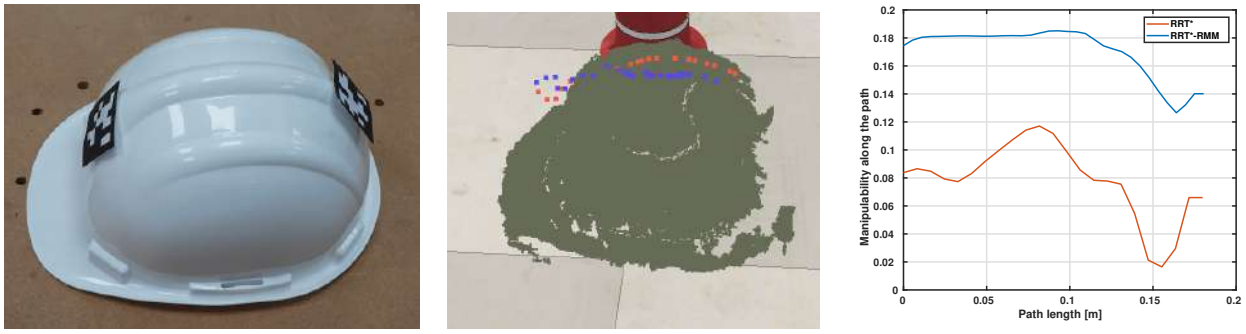
Fig. 2 shows the experimental setup with a cylindrical container emulating a nuclear waste barrel. A heat map overlaid on the surface of the object represents the manipulability corresponding to the configuration the robot is in when its EE is at the point on the object. We use the standard inverse kinematics (IK) of the robot to compute the joint configurations. We developed a full ROS package to compute the optimal cutting path. As the robot URDF can be loaded onto the parameter server and used by the algorithm, we can easily repeat our computation with any manipulator whose URDF is available.

We used four objects (Fig. 6) to illustrate the effectiveness of our approach in generating a cutting path on different objects surfaces. These objects are a barrel (cylindrical container), a curved object (made of foam), a safety helmet and a flat object (also made of foam). These objects represent typical objects that are to be cut in a nuclear environment. Future work will include the deployment of the proposed algorithm to cut real-world object found in such environments.

Fig. 5 shows the helmet we used for our experiments along with the markers attached to the object. The markers fix the initial and end point of the cutting path. These points can be

[2]Experimental results are reported in the attached video.

[3] Although the algorithm needs the robot kinematics, our ROS implementation takes the robot URDF directly from the ROS server parameter. Therefore, we present some data collected with Panda and some others with Sawyer to show the robustness of our approach to changes of the kinematic chains.



(a) (b) (c)

Fig. 5. This image shows the path proposed by the two algorithms for a path connecting the two points indicated with the markers on a helmet in 5(a). The helmet has 3 ridges on its shell. The top part of the helmet consists of smooth surfaces with large curvatures, whereas the shell and the ridges are connected with very small curvature. 5(b) shows the point cloud of the helmet captured by the camera in V-REP and the path proposed by RRT* and our proposed approach are shown in red and blue dotted lines, respectively. In 5(c), the manipulability of both paths is shown.

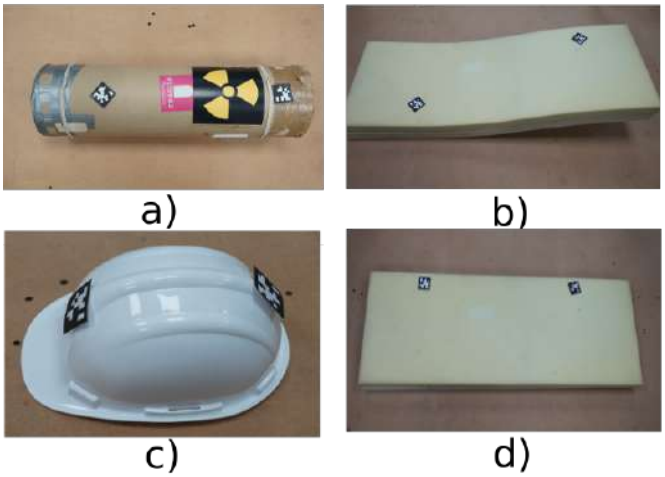


Fig. 6. Test objects used for testing our algorithm, (a) a barrel, (b) a curved object, (c) a helmet and (d) a flat object. The pictures also show the position of the markers.

provided by a human operator during real-world deployments. Fig. 5(b) shows the point cloud of the helmet captured by the camera and visualised in V-REP. The paths computed by RRT* and our proposed approach, RRT*-RMM, are shown with red and blue dotted lines, Fig. 5(b). These results show that RRT* and RRT*-RMM effectively generate cutting path on the object surface. Fig. 5(c) also shows the manipulability corresponding to the paths obtained by RRT* and RRT*-RMM with red and blue lines, respectively. This figure shows that our algorithm finds a path that has a significantly improved manipulability for the robot throughout the whole path.

We see that RRT* generates a path specific just to the shape of the object. In contrast, RRT*-RMM computes paths not only specific to the shape of the object, but also specific to (i) the position of the object relative to the robot base frame and (ii) the kinematic chain of the robot. If we change either object position or the robotic arm, RRT*-RMM computes another path which is best for that scenario. As such, our algorithm always finds a path which is the best fit for the specific problem setting.

We performed similar experiments with all four objects shown in Fig. 6. Sample PC of 4 objects visualised in V-REP are shown in Fig.7 along with the computed path by RRT* and RRT*-RMM, red and blue respectively. In detail, Fig. 7(d) shows that the robot faces singularity if it follows the path obtained by RRT* for the flat object shown in Fig. 6. In contrast, it does not experience this issue when using the path obtained with RRT*-RMM because the approach is explicitly designed to avoid such an issue. The paths visualised in V-REP in Fig. 7 correspond to the markers positions shown in Fig. 6. These figures show that slight differences in the path obtained by RRT* and RRT*-RMM yield higher manipulation capability for the manipulator.

Because the core of our planning algorithm is basically random, we need a statistical evidence that our approach performs better. Therefore, we repeated the experiments five times, each time with a different endpoint for every object and compute cutting path by RRT* and RRT*-RMM. We collected the data of manipulability and the length of the computed paths by RRT* and RRT*-RMM and the corresponding box plots are shown in Figs. 8 and 9. The object surfaces we used are largely dissimilar, *e.g.* we have objects with different geometry such as a flat surface and a helmet with sphere-like geometry. Fig. 8 shows the box plots of obtained manipulability for objects shown in Fig. 7. The mean and variation of the data summarised in the box plots in Fig. 8 suggest the that RRT*-RMM yields paths with generally higher manipulability. Although the path computed by our algorithm and RRT* differs at each repetition due to (1) the randomness of node generation and (2) the chosen goal point, Fig. 8 shows our approach yields bigger manipulability values in a mean sense. RRT*-RMM also yields smaller variation of manipulability which is another desired characteristic of our algorithm. As we expected, the path length is increased with respect to the RRT* baseline since it is leveraged for a higher manipulability throughout the path. Nonetheless, this increment is not very high as it is shown in Fig. 9. The increased path length is $\sim$10% for the barrel, the curved object and the safety helmet, and $\sim$50% for the flat object.

In this paper, we do not address the lower-level dynamics and control issues of the forceful contact between robot and cut material, while a cutting path is being followed. A forthcoming

(a) (b) (c) (d)

Fig. 7. Paths proposed by RRT* and RRT*-RMM for the four objects, using initial and target positions shown in Fig. 6 using the markers. Fig. 7(d) shows how RRT* proposes a path close a singularity for the robot (orange path) instead, the RRT*-RMM does a semicircle route to avoid it. In our experiments, we empirically selected $\alpha = 0.7$ to trade off path's length and manipulability.

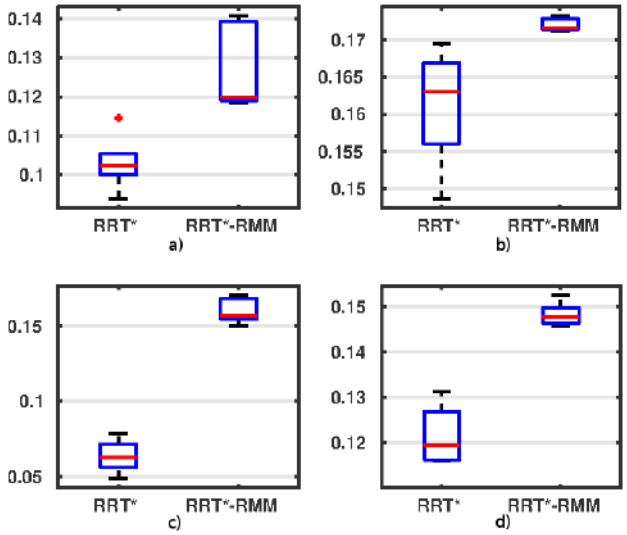


Fig. 8. Box plot of the manipulability values obtained by RRT* and RRT*-RMM for all four objects. The order of these figures corresponds to the order of object figures shown in Fig. 6, a) the barrel, (b) the curved object, and (d) the flat object.

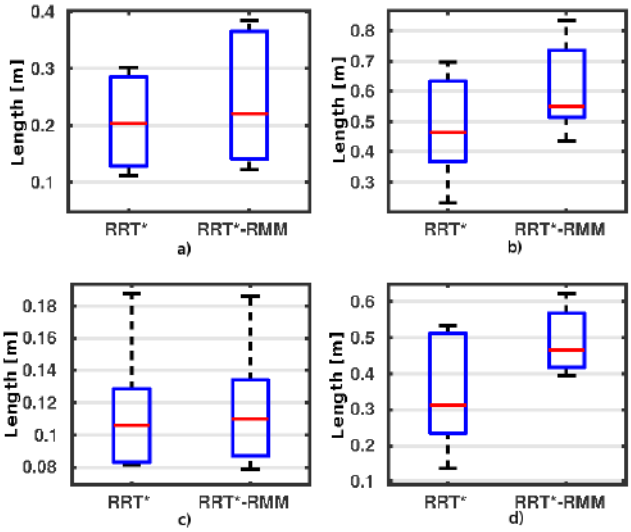


Fig. 9. This figure shows the path length found for the same objects. Every object has been tested using five goal positions, and the plots include the data for all the trials. The order of these figures corresponds to the order of object figures shown in Fig. 6, a) the barrel, (b) the curved object, and (d) the flat object.

paper will show how this can be addressed using hybrid force-position control, building on our previous work [30]. In contrast, this paper focuses on the higher-level problem of planning a safe cutting path for the manipulator, on a surface obtained by a vision system. In our experiments, we inflated the Point Cloud, *i.e.*, we modified PC to surround the object surface with 0.02 [m] off-set. Therefore, our algorithm computes a path safe for our real robot demonstrations, *i.e.*, the robot could move along the obtained path while keeping a 0.02 [m] distance to the object's surface to avoid forceful interaction with the object.

Our cost function may be applied as the objective function of any kind of numerical optimiser. However for proof of principle, here we demonstrate how to incorporate it into RRT*, and it can be readily extended to other common path planners, *e.g.*, RRT, PRM. We showed our approach can successfully yield the cutting paths using two different 7 dof arms. Nonetheless, it can be easily applied to any dof manipulators.

In this paper, we consider only the surfaces with different shapes, which are smooth (Riemannian) manifolds. Although object with discontinuities or non-smooth objects cannot be directly tackled by our approach, we can address these problems by considering a sequence of sub-problems over continuous and smooth surfaces. The final cutting path can then be the union of the solutions to the sub-problems. The choice of the parameter α in eq. 10 allows us to weigh the manipulability component of the algorithm based on the cutting task we are performing, *e.g.* cutting with knife and rotating tool may need different manipulation capabilities. Although a value of α close to 1 might seems a reasonable option, it has the shortcoming of resulting in non-desirable long and very jerky cutting paths. In our experience, values below 0.85 lead to cutting paths with an acceptable length and increased manipulability along the path.

This paper is accompanied by a video including experiments using Sawyer robot in V-REP simulation and using real Panda robot.

VI. Conclusion

This paper addresses the problem of constrained motion planning from vision, which enables a robot to move its end-effector on an observed surface, given start and destination points. We find robot trajectories which maximise the robot's manipulability throughout the motion. This work has application in industrial problems of robotic rough cutting. Our approach uses a mapping between Euclidean space and Riemannian manifold to project the random samples taken from the Point Cloud onto the object surface. This mapping step in our algorithm allows us to compute the path on an object surface using only a Point Cloud, without the need of a complete 3-D model. Moreover, we use a cost composed of the sum of manipulability indices along the path and the distance travelled between the initial and evaluated point. We incorporated this cost into the RRT*. The manipulability index added to the RRT* cost assures the generated paths to yield higher manipulability values.

We presented a series of experiments with a Panda robot and a Sawyer robot. The experiments include computing the cutting paths on 4 different objects. Since the core of RRT* and RRT*-RMM is random sample generation, we perfomed a statistical study that shows how RRT*-RMM improves the manipulability index while trading off the length of the cutting path, *i.e.*, RRT*-RMM obtains an increased manipulation capability (avoiding robot-related issues) at the cost of increased path length.

Future work includes the extension of this algorithm to non-Riemannian surfaces as this would allow the application of this algorithm to objects with sharp edges and discontinuities. Moreover, we are studying a suitable control architecture to include the proposed algorithm in a force control framework, to enable an effective implementation of forceful cutting operations.

References

[1] N. Marturi, A. Rastegarpanah, V. Rajasekaran, V. Ortenzi, Y. Bekiroglu, J. Kuo, and R. Stolkin, "Towards advanced robotic manipulation for nuclear decommissioning," in *Robots Operating in Hazardous Environments*, IntechOpen, 2017.

[2] M. Talha, E. A. M. Ghalamzan, C. Takahashi, J. Kuo, W. Ingamells, and R. Stolkin, "Towards robotic decommissioning of legacy nuclear plant: Results of human-factors experiments with tele-robotic manipulation, and a discussion of challenges and approaches for decommissioning," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 166–173, 2016.

[3] "Nuclear provision: the cost of cleaning up britains historic nuclear sites," *UK Nuclear Decommissioning Authority*, 2019.

[4] J. M. Dodds and J. Rawcliffe, "Radionuclide distribution during ytterbium doped fibre laser cutting for nuclear decommissioning," *Progress in Nuclear Energy*, vol. 118, p. 103122, 2020.

[5] M. Selvaggio, A. Ghalamzan E., R. Moccia, F. Ficuciello, B. Siciliano, *et al.*, "Haptic-guided shared control for needle grasping optimization in minimally invasive robotic surgery," in *IEEE/RSJ International Conference Intelligent Robotic System*, 2019.

[6] L. Peternel, N. Tsagarakis, and A. Ajoudani, "Towards multi-modal intention interfaces for human-robot co-manipulation," in *IEEE/RSJ international conference on intelligent robots and systems*, pp. 2663–2669, IEEE, 2016.

[7] R.-S. Lin and Y. Koren, "Efficient tool-path planning for machining free-form surfaces," *Journal of engineering for industry*, vol. 118, no. 1, pp. 20–28, 1996.

[8] Amir M. Ghalamzan E., N. Mavrakis, M. Kopicki, R. Stolkin, and A. Leonardis, "Task-relevant grasp selection: A joint solution to planning grasps and manipulative motion trajectories," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 907–914, 2016.

[9] A. Ghalamzan, F. Abi-Farraj, P. R. Giordano, and R. Stolkin, "Human-in-the-loop optimisation: mixed initiative grasping for optimally facilitating post-grasp manipulative actions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems f*, pp. 3386–3393, IEEE, 2017.

[10] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, pp. 396–404, Springer, 1986.

[11] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," 2009.

[12] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*, pp. 4569–4574, IEEE, 2011.

[13] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 113–120, IEEE, 1996.

[14] L. Vinet and A. Zhedanov, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Tech. Rep. 8, 2011.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[16] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Intelligent Robots and Systems*, pp. 2997–3004, IEEE, 2014.

[17] D. Kruger, R. Stolkin, A. Blum, and J. Briganti, "Optimal auv path planning for extended missions in complex, fast-flowing estuarine environments," in *IEEE International Conference on Robotics and Automation*, pp. 4265–4270, IEEE, 2007.

[18] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras, "Evolutionary algorithm based offline/online path planner for uav navigation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 33, pp. 898–912, Dec 2003.

[19] Y.-C. Lin and D. Berenson, "Humanoid navigation planning in large unstructured environments using traversability - based segmentation," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 7375–7382, 2018.

[20] B. Kim, T. T. Um, C. Suh, and F. C. Park, "Tangent bundle RRT: A randomized algorithm for constrained motion planning," *Robotica*, 2016.

[21] L. Han and N. Amato, "A kinematics-based probabilistic roadmap method for closed chain systems: Li han, texas a nancy m. amato, texas a," in *Algorithmic and Computational Robotics*, pp. 243–251, AK Peters/CRC Press, 2001.

[22] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, 2011.

[23] L. Jaillet, J. Cortés, and T. Siméon, "Transition-based RRT for path planning in continuous cost spaces," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2145–2150, 2008.

[24] L. Jaillet, J. Corts, and T. Simon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, pp. 635–646, Aug 2010.

[25] C. Urmson and R. G. Simmons, "Approaches for heuristically biasing rrt growth.," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1178–1183, IEEE, 2003.

[26] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *IEEE International Conference on Robotics and Automation*, pp. 1874–1879, 2006.

[27] E. K. Lavrovsky, M. Vidyasagar, K. Krishnamurthy, T. Asano, Y. Sakawa, A. Tzes, A. G. Ulsoy, R. A. Scott, Y. Wu, N. Xi, A. Isidori, D. H. Young, W. Weaver, L. Lanari, and G. Ulivi, "Using Manipulability to Bias Sampling During the Construction of Probabilistic Roadmaps," vol. 19, no. 6, pp. 1020–1026, 2003.

[28] T. Yoshikawa, "Manipulability of robotic mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[29] G. Leonor and J. Natrio, "An introduction to riemannian geometry," *Universitex, ISBN 9783319086668*, 2014.

[30] V. Ortenzi, R. Stolkin, J. A. Kuo, and M. Mistry, "Projected inverse dynamics control and optimal control for robots in contact with the environment: A comparison," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4009–4015, IEEE, 2015.